The Command System

By modifying the data contained in the application shell system files, you can define the behavior and appearance of systems you develop with the frame gallery. Because this information is stored centrally in files, it can be easily updated. Handling for multiple-language applications and access protection can be easily maintained using the administration system provided in the application shell instead of being programmed explicitly within all dialogs.

The following topics are covered below:

- Information Objects and Application Components
- Data Buffer
- Access Protection
- Command Processor
- Command Processing Description

Information Objects and Application Components

The following table lists the application components influenced by the specific information objects:

Information Object	Application Component	Used to Define
Command	Menu item	Name DIL-text
	Tool bar item	Bitmap DIL-text
	Bitmap	DIL-text
	Push button	Possible Name DIL-text
	Direct Call	Name (action)
Object Type	Icon-based Navigation	Name DIL-text
	Direct Call	Name
Function	Icon-based Navigation	Name DIL-text
	Main dialog for function	String
Application	Icon-based Navigation	Name DIL-text
	MDI-frame	String

Data Buffer

The most frequently used data are read into main storage at the start of an application, thereby reducing the number of database accesses to the application shell system files as well as reducing the overall network overhead.

This storage area (data buffer) is implemented as a transparent background dialog.

In order to access data in the data buffer, dialogs send requests (events) to this background dialog, which responds via a parameter interface.

The following data are contained in the data buffer:

Information Object	**Data Fields**
Commands (up to 400)	Command ID
	Command Type (main and subtype)
	Name
	DIL-text
	Bitmap
	Parameter
Tool bars (up to 70)	Tool bar ID (up to 30 tool bar items (commands))
Dialog Types (up to 70)	Dialog Type ID
	Tool bar ID
Object Types (up to 250)	Object Type ID
Functions (up to 800)	Function ID
	Up to 3 actions (command from main type action)
	Object ID
Applications (up to 80)	Application ID

Access Protection

Access protection is provided at the function level. A function is either permitted or prohibited for a given user.

The access to object types and applications is controlled by the access protection at the function level.

An object type (and its listing) is only permitted if at least one permitted function exists for this object.

An application is only permitted if at least one of its children (application, function, object type) is permitted.

This means that a user can only see what is authorized via the functional access protection. This has the following implications within an application:

- Menu items, tool bar items, bitmaps and push buttons, which are associated with a command, will be omitted if they are not permitted for a user. Only commands from the main type Action, Application Start, List Start and Function Start are affected by access protection.

Command Type	Prohibited
Action	Function is not permitted for current object type and action
Function Start	Function is not permitted
List Start	Object type is not permitted
Application Start	Application is not permitted

- Submenus are omitted for which no permitted command exists.
- The graphical navigation only offers the applications, functions and object types for selection which are permitted.
- The "Direct Call" dialog offers only object types for which at least one function is permitted.
- The "Direct Call" dialog displays only actions (commands from main type action) for which a function for the selected object type is permitted.

Access protection can be enabled or disabled for an entire application or for individual users.

Access Protection for Application	Access Protection for User	Check Authorization for Functions
Yes	Yes	Yes
Yes	No	No
No	Yes	No
No	No	No

Undefined users can only start an application if access protection for the application is disabled. Only those functions are checked for which access protection is enabled. Functions without access protection are available to all users.

If authorizations for functions are to be determined, the ordering of functions to function groups and the assignment of function groups to users should be considered. A function group can have permitted and/or prohibited functions.

This and the access protection defaults for the application thereby provide the set of permitted functions for a user.

Default Value of the Application	Set of Permitted Functions
All functions permitted	All functions of the application, except those functions which are prohibited via a function group. Plus those functions which are permitted via a function group.
All functions prohibited	All functions which are permitted via a function group, except functions which are prohibited via a function group, plus functions without access protection.

During start-up of the application, all permitted functions are determined based on information in the data buffer.

This in turn determines the permitted object types (see above), which are placed into the data buffer.

Using this information, the application structure is checked and the permitted applications (see above) are placed in the data buffer.

All commands of the application are read into the data buffer, whereby prohibited commands (Function Start, Application Start, Object Start) are marked.

Whether or not an action is permitted can only be determined at runtime. The action will be permitted only if the action and the current object combination referenced in the current dialog is contained in the data buffer.

Command Processor

The command processor is a background dialog which controls the processing of defined commands. Command are activated via menu items, tool-bar buttons, push buttons, and icons with which a command ID has been associated.

The dialog communicates with the command processor via events.

During the initialization of a dialog, the following tasks are executed:

- If the value of the variable LZ_SECURITY is True, it is checked whether the commands attached to the dialog elements are permitted. If a command is prohibited, the corresponding dialog elements are deleted.
- Depending on the value of the variable LZ_STRING_REPLACE, the STRING- and DIL-attributes are assigned the corresponding values of the command entries stored in the data buffer.

The tasks below are dialog element specific:

Menu Items

Empty submenus and superfluous separators are removed.

Tool Bar Items

The BIT-MAP-FILE-NAME for the dynamically added tool bar items are read from the data buffer. Superfluous separators are removed.

If the tool bar items are defined in the dialog editor and a dialog type is assigned to the dialog, the corresponding tool bar items are added dynamically behind the existing items.

Bitmaps

The BIT-MAP-FILE-NAME is read from the data buffer.

For the application programmer, the command processor is the most significant interface for controlling dialog elements with command ordering.

If, for example, a command must be disabled, it is only necessary to indicate this to the command processor. The associated dialog elements are then determined and disabled by the command processor.

The following functions for commands and the associated dialog elements are available:

Function	**Affected Dialog Element**
Enable	Menu item, tool bar item, push button, bitmap
Disable	Menu item, tool bar item, push button, bitmap
Set marker	Menu item
Remove marker	Menu item
Rename	Menu item, push button
Replace (Attribute String)	Menu item, push button
Replace (:n:) position holder	
Replace (attribute DIL-TEXT)	Menu item, tool bar item, push button,
Replace (:n:) position holder	bitmap
Delete	Menu item, tool bar item, push button, bitmap

Command Processing Description

This section describes the processing of dialog elements which are assigned to a command.

Command processing consists of the following steps:

1. Assignment of commands to dialog elements via the attribute COMMAND-ID.
2. Clicking of a dialog element which is assigned to a command.
3. Retrieval of associated command data via the command processor.

Command	Variable
ID	LZ_FRAME_CMD_ID
Main type	LZ_FRAME_CMD_TYPE_MAIN
Sub type	LZ_FRAME_CMD_TYPE_SUB
Parameter	LZ_FRAME_CMD_PARM

4. Call to user subroutine Z_CMD_EXEC_START.
   Additional processing prior to standard processing is possible here, for example, to change the command.
5. Standard processing Z_CMD_EXEC_FRAME.
   Standard processing of the command. If the command cannot be processed by frames, the user subroutine Z_CUSTOM_CMD is called.
6. Call to user subroutine Z_CMD_EXEC_END.
   Additional processing subsequent to standard processing can be performed at this point.

